Une approche générique pour l'adaptation dynamique des IHM au contexte

Safa Hachani, Sophie Dupuy-Chessa, Agnès Front

Une approche générique pour l'adaptation dynamique des IHM au contexte

Safa Hachani , Sophie Dupuy-Chessa et Agnès Front

Université de Grenoble, CNRS, LIG
BP 72, 38402 Saint Martin d'Hères Cedex, France
{ Prenom.Nom@imag.fr }

RESUME
Les contextes d'usage se diversifient. Il devient alors nécessaire d'adapter les Interfaces Homme-Machine (IHM) au contexte. Dans cet article, nous adoptons une approche basée sur les modèles pour l'adaptation des IHM au contexte. Notre approche s'appuie sur une spécification générique et adaptable des modèles de tâches. Cette spécification considère simultanément les similarités et les variations existantes entre différents contextes d'utilisation d'une même application. De tels modèles sont ensuite ajustés par transformation de modèles à l'instar de l'Ingénierie Dirigée par les Modèles (IDM [6]), à la situation d'utilisation.

MOTS CLES : Modèle de tâches, Contexte d'utilisation, Adaptation, Variabilité, IDM.

ABSTRACT
The contexts of use vary a lot. It becomes fundamental to adapt user interfaces (UI) to the context. In this article, we choose an approach based on models for UI adaptation. Our approach facilitates the adaptation through a generic and flexible specification of the task tree model. This specification takes into account both similarities and existent variations between different users contexts of the same application. Such models are afterwards adjusted by model transformation such as the Model Driven Engineering (MDE [6]), according to the user situation.

CATEGORIES AND SUBJECT DESCRIPTORS:
User Interfaces; H.5.2 [Information Interfaces and Presentation]: User Interfaces|User-centered design.

GENERAL TERMS: Design, Human Factors

KEYWORDS: Task tree model, Context of use, Adaptation, Variability, MDE.

IHM 2009, 13-16 Octobre 2009, Grenoble, France

INTRODUCTION

Les progrès technologiques tels que la miniaturisation des capteurs, et l'explosion des technologies de communication permettent aux utilisateurs d'accéder aux informations n'importe où et n'importe quand. Les contextes d'usage, définis par un triplet <utilisateur, plate-forme, environnement> [3] se diversifient et les utilisateurs attendent que leurs applications s'adaptent à ces contextes : par exemple, un utilisateur nécessitant des soins médicaux, veut connaître les médecins pouvant le recevoir ; l'application devra afficher les médecins à proximité (caractéristiques de l'environnement) sur son téléphone portable (caractéristique du dispositif) dans la langue la plus appropriée (caractéristique de l'utilisateur). Une telle adaptation des Interfaces Homme-Machine (IHM) à la situation contextuelle devient indispensable, si bien que les travaux de recherche dans ce domaine connaissent un essor important.

Parmi les approches proposées, nous choisissons une approche basée sur les modèles qui permet 1) de capitaliser les connaissances d'adaptation, en les représentant explicitement sans les noyer dans le code ; 2) d'estomper la frontière entre conception et exécution du système en considérant les modèles vivants à l'exécution. Chaque modèle (du modèle de tâches au code) décrit une perspective du système [13] qui est adapté au contexte. Adapter un système interactif revient alors à adapter ses modèles.

Dans cet article, nous proposons d'adopter une approche générique qui facilite la mise en œuvre de l'adaptation des IHM au contexte en capitalisant une même logique d'adaptation applicable quelles que soient les particularités des contextes d'utilisation ou d'applications. Il s'agit d'une approche basée sur la variabilité et l'Ingénierie Dirigée par les Modèles (IDM [6]) que nous appliquons au modèle de tâches. Les concepteurs pourront désormais, à l'aide de notre proposition, concevoir des modèles de tâches à variantes adaptables suivant les divers contextes. Pour mettre en œuvre cette approche, nous introduisons dans le modèle de tâches la notion de variabilité. Le concept de *variabilité* permet de factoriser les parties communes à toutes les situations contextuelles tout en identifiant les parties variables. Les variantes sont ensuite réduites par transformation de modèles suivant la situation contextuelle.

Cet article est organisé comme suit. La première section définit la problématique de l'adaptation des IHM au contexte et présente une étude des approches existantes. La deuxième section présente notre approche et l'illustre par un exemple. La section suivante détaille notre méta-modèle pour représenter un contexte de façon générique, ainsi que le méta-modèle de tâches adaptable et les règles de transformation utilisées pour l'adaptation. La dernière section énonce la conclusion de ce travail.

APPROCHES EXISTANTES ET PROPOSITIONS D'APPUI

Cette section a pour objectif de cerner l'espace problème de l'adaptation des IHM et de présenter les approches existantes. Nous utilisons pour l'illustration un scénario d'interaction qui servira aussi pour avancer nos propositions.

Scénario-Type d'Interaction

Ce scénario est l'un des cas d'étude du projet européen UsiXML et montre par l'exemple différents cas d'adaptation au contexte d'interaction.

"Un couple grenoblois est en voyage à Turin. Pierre montre des signes alarmants de santé. Personne à qui se renseigner. Ashley utilise l'application Compose installée sur son SmartPhone. Elle demande en anglais, sa langue maternelle, de lui trouver une solution. Compose lui propose alors plusieurs solutions : le médecin de garde ; les urgences à l'hôpital le plus proche, etc. On opte pour le médecin de garde. Compose assemble alors une Interface Homme-Machine permettant d'appeler le médecin (affichage du numéro). Un plan est affiché pour s'y rendre. La pharmacie de garde est également indiquée et localisée. Ashley fait migrer la carte de navigation sur le PC de Pierre le temps d'appeler le médecin. Ainsi, en parallèle, Pierre peut prendre connaissance des lieux. "

Ce scénario illustre les besoins d'adaptation du système. Dans cet article, nous ne traitons que les besoins relatifs à la langue et au dispositif d'interaction : 1) Ashley doit pouvoir interagir selon ses préférences de langue (Français ou Anglais) ; 2) Ashley interagit avec le système via son SmartPhone, caractérisée par sa taille d'écran réduite. Cet exemple se base donc sur des éléments de contexte assez simples afin d'illustrer de manière pédagogique notre proposition.

Approches d'Adaptation des Systèmes Interactifs aux Contextes : Limites et Recommandations

L'objectif commun des approches existantes pour l'adaptation au contexte est la capitalisation du savoir et du savoir faire (i.e. comment adapter). Elles se divisent en deux catégories : les approches basées sur les composants [2] [4], et celles basées sur les modèles [7] [10] [11] [12] [13].

Les approches basées sur des composants permettent l'adaptation des systèmes interactifs par transformation dynamique d'assemblages de composants, qui sont prédéveloppés pour des besoins spécifiques. L'ajout ou le retrait de composants, ainsi que la création ou la suppression de liaisons entre composants, permettent alors de construire divers systèmes interactifs. Cette approche résout bien le problème de l'adaptation dynamique, mais elle présente l'inconvénient d'utiliser un ensemble de variantes fixes et prédéfinies pour diverses utilisations.

Au contraire, les approches basées sur les modèles expriment un large panel de variables au travers des transformations de modèles explicites [13] ou implicites [12] pour le concepteur. En particulier, l'IDM met à disposition des concepts, des langages et des outils pour créer et transformer des modèles en se basant sur leur méta-modèle [6]. L'approche [13] permet ainsi, à travers des transformations de modèles, l'expression de plusieurs variantes d'interfaces, à partir d'une seule spécification qui met en relation le contexte avec les tâches et les interacteurs. L'approche [7] permet de faciliter l'implémentation de l'adaptation par insertion dans le code du système interactif de points d'observation, des changements de contexte d'usage, et d'actions à exécuter pour l'adaptation. Toutefois, cette approche encapsule les savoirs d'adaptation dans le code alors que nous cherchons à les rendre plus accessibles aux non-développeurs.

Pour ce faire, nous considérons que l'approche [13] répond au plus près à certains de nos objectifs, à savoir faciliter l'adaptation des IHM en nous basant sur des modèles qui représent les savoirs, vivants à l'exécution. Il reste à faciliter la mise en œuvre des savoirs-faire par des transformations génériques et donc réutilisables.

Approche de Sottet. Dans l'approche de Sottet, un concepteur doit définir un module de transformation par paramètre du contexte. Il s'agit d'un ensemble de règles spécifiques. Par exemple, des règles permettant selon les caractéristiques des utilisateurs, de générer des interfaces en français ou en anglais. Avec cette approche (cf. Figure 1), la conception des transformations est non générique et donc non applicable à tous les contextes d'utilisation. Ainsi, si de nouveaux besoins de personnalisation se présentent (tel le contexte 2 sur la figure), une extension du module de transformation est nécessaire. Il faut donc écrire de nouvelles règles de transformation, ce qui n'est guère envisageable concrètement.

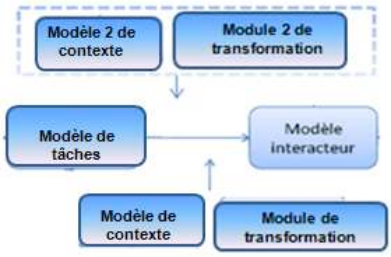


Figure 1 : Transformations spécifiques à une application et un contexte particulier

Cette approche présente ainsi des limites. Elle permet le développement d'un seul cadre spécifique de contextes, et donc de transformations particulières. Elle utilise ainsi une logique pour une seule application et ses divers contextes d'usage préconçus, mais n'offre pas un même savoir faire pouvant s'appliquer à toute application et à tout contexte.

Proposition d'appui

Comme dans [13], nous souhaitons proposer une approche basée sur les modèles et les transformer suivant la situation contextuelle. Mais nous souhaitons 1) adapter les tâches et pas les interacteurs, 2) proposer des règles de transformations génériques pour que les concepteurs n'aient pas ce travail difficile à effectuer. Pour ce faire, nous devons étendre les modèles de tâches pour les rendre adaptable au contexte. Nous avons choisi une extension des arbres de tâches avec des points de décision [11] que nous définissons comme des points de variation suivant le principe de variabilité introduit pour les « lignes de produits » [8]. Ainsi les sous-arbres d'un point de variation ont une structure particulière. La variabilité est appliquée au modèle de tâches pour permettre de proposer des règles d'adaptation génériques. Nous proposons de définir et d'utiliser pour ce faire, un méta-modèle générique de contexte et un méta-modèle de tâches adaptable.

VERS UNE APPROCHE GENERIQUE POUR L'ADAPTATION DES IHM AU CONTEXTE

Le but de notre travail est d'offrir une approche générique adaptant, de la même manière, différentes IHM sensibles au contexte. Pour cela, nous proposons de capitaliser le savoir faire d'adaptation dans un module de transformation générique et réutilisable qui s'appliquera identiquement sur les divers modèles de tâches sources des applications.

Cadre Général de l'Approche

Nous illustrons dans ce qui suit la cartographie de notre approche (cf. Figure 2). Nous nous basons sur et utilisons d'une part un méta-modèle de tâches cible spécifique (déjà proposé par Sottet [13]), et définissons deux nouveaux méta-modèles sources qui sont les suivants :

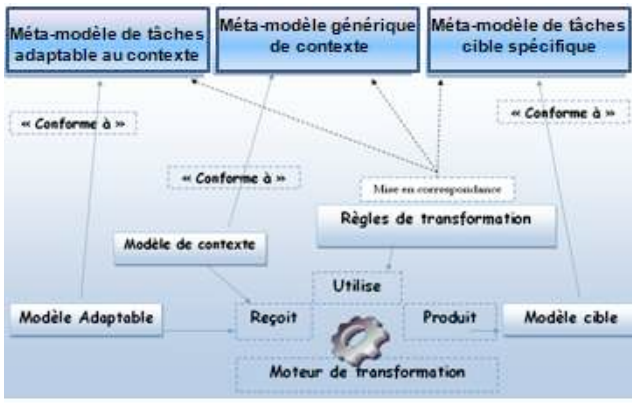


Figure 2 : Cartographie de l'approche proposée

- Un méta-modèle générique de contexte qui propose une description formelle des éléments de contexte et leurs paramètres.

- Un méta-modèle de tâches adaptable au contexte, supportant la variabilité et exprimant le lien avec les éléments du contexte.

La variabilité est définie comme étant « la capacité d'un système à être changé, adapté et configuré en fonction d'un contexte spécifique » [18]. L'idée principale consiste à identifier ce qui est fixe et ce qui est variable, en fonction des caractéristiques des éléments du contexte. Ainsi, l'abstraction (première étape de notre approche) permet d'identifier les comportements similaires entre plusieurs contextes d'utilisation et ceux variables. Les comportements variables sont représentés dans des *points de variation* [5] aussi appelés points de décision dans [11]. Il s'agit d'emplacements où des choix seront faits pour identifier les différentes manières (i.e. l*es variantes*) d'accomplir le point de variation.

Les différents points de variation sont réduits en fonction de la situation contextuelle. Cette réduction est effectuée de règles de transformation génériques, c'est-à-dire qui restent inchangées quelque soit l'application ou le contexte. Une transformation peut être vue comme un ensemble de règles qui, prises ensemble, décrivent comment un ou des modèles sources sont transformés en modèles cibles. Une transformation fait ainsi correspondre pour chaque concept du méta-modèle de tâches adaptable au contexte un concept du méta-modèle de tâches cible. Ces règles de transformation se basent sur les méta-modèles à partir desquels sont exprimés les modèles mis en jeu. L'adaptation se fait sur les instances des deux méta-modèles sources pour fournir un modèle spécifique à un contexte particulier et instance du méta-modèle cible adopté.

Illustration de l'Approche

Nous commençons par l'identification des différentes variantes par rapport aux besoins d'adaptation déjà identifiés (cf. section Scenario Type d'Interaction), pour pouvoir concevoir le modèle de tâches adaptable au contexte. Nous appliquons sur le modèle de tâches adaptable, en utilisant un modèle de contexte, les règles génériques de transformation présentées ultérieurement afin d'avoir un modèle de tâches correspondant à une situation contextuelle.

Abstraction : Modèle de tâches adaptable au contexte. Afin de construire notre modèle de tâches adaptable, nous commençons par l'identification des points de variations par rapport aux besoins d'adaptation (i.e. attributs du contexte).

L'identification de la variabilité est liée à la définition des points de variation, qui représentent ce qui peut différer d'une situation contextuelle à une autre, et leurs variantes (cf. figure 3). Cette phase se base aussi sur l'association de chaque point de variation aux attributs du contexte desquels il dépend, et des variantes aux contenus qui les déterminent. Pour le scénario étudié, on distingue :

- Une tâche point de variation alternative « spécifier demande » liée au paramètre de contexte « Langue ». On peut avoir deux variantes possibles de cette tâche : une variante pour les utilisateurs qui préfèrent la langue Français et une pour ceux qui préfèrent la langue Anglais.

- Une tâche point de variation optionnelle « migration de carte » liée au paramètre « TailleEcran ». Cette tâche optionnelle fait partie de la réduction si le dispositif utilisé lors de l'interaction admet un écran de taille réduite (SmartPhone, PDA, etc.).

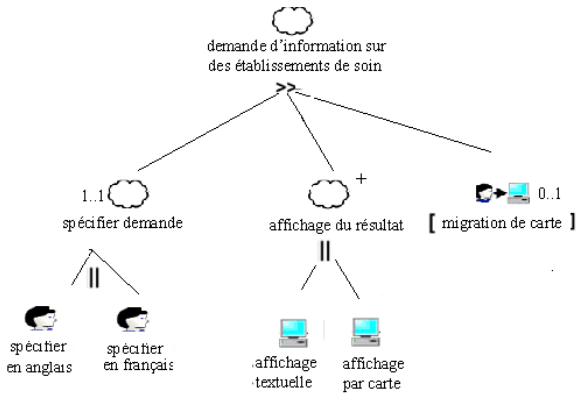


Figure 3 : Modèle de tâches adaptable

Les tâches point de variation sont représentées par leurs cardinalités associées (1..1, 0..1) qui seront précisées en section suivante. On peut noter que contrairement à ce qui est présenté dans [11] et [14], les points de variation ont pour sous-arbre des variantes avec des possibilités de choix explicitées.

Réduction de la Variabilité : Adaptation par transformation de modèles. Lors de la détection d'un changement de contexte, le système s'adapte. Pour le sénario étudié, Ashely se trouve dans une situation caractérisée par une contrainte de langue (Anglais) et une deuxième contrainte concernant la taille réduite de son Smartphone (cf. Figure 4).

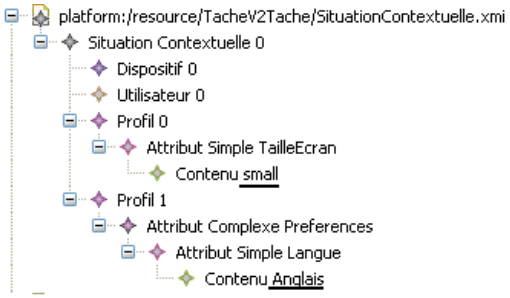


Figure 4 : Modèle de contexte 1

Face à ces besoins de personnalisation, le processus de réduction de la variabilité, basée sur les transformations génériques, permet l'adaptation du modèle de tâches. Le module de transformation utilise en entrée le modèle de tâches adaptable (cf. figure 3) et le modèle de contexte (cf. figure 4) sur lesquels s'appliquent les règles de transformation génériques tout en se basant sur leurs méta-modèles respectifs.

Le paramètre de contexte « TailleEcran » duquel dépend la tâche point de variation optionnelle « migration de carte » est actif dans la situation contextuelle en cours. Son contenu « small » correspond à la condition de sélection de la tâche point de variation « migration de carte ». Cette tâche fait donc partie de la réduction. Cette réduction de la variabilité donne lieu au modèle de tâches spécifique au contexte présenté en figure 5.

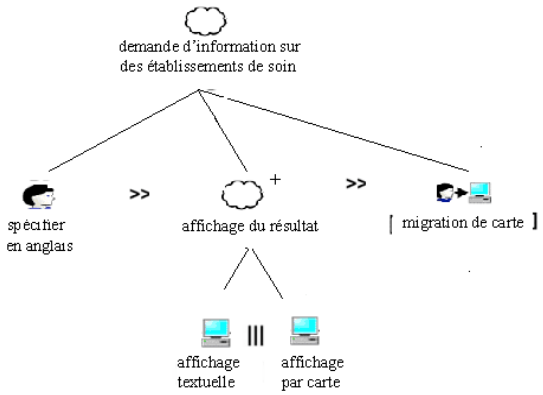


Figure 5 : Modèle de tâches spécifique au contexte 1

Un deuxième utilisateur utilisant aussi son smart-Phone mais préférant le français peut interagir avec le système. Ce changement du contexte induit une adaptation du système par l'exécution du module de transformation : la tâche point variation « spécifier demande » du modèle de tâches adaptable est réduite à la tâche variante « spécifier en français » correspondant à la préférence de l'utilisateur en langue. Le modèle de tâches obtenu est donc très semblable à celui de la figure 5, mis à part que la tâche « spécifier en anglais » est remplacée par « spécifier en français ».

LES TROIS PILIERS DE NOTRE APPROCHE

Dans la section précédente, nous avons illustré l'apport de notre proposition. Cette section détaille les trois piliers de notre approche : le méta-modèle de contexte générique, celui de tâches adaptable au contexte et les règles de transformation génériques.

Méta-Modèle de Contexte Générique

Contrairement à une approche de concrétisation des modèles de tâches qui se basent sur des patterns pas forcément explicites pour les concepteurs [12], nous avons choisi de coupler le modèle de tâche variable avec un modèle de contexte. Ce modèle de contexte doit être facile à représenter et doit pouvoir prendre en compte la diversité des situations contextuelles. Ceci crée le besoin d'avoir un méta-modèle de contexte qui soit à la fois générique, flexible et réutilisable qui intègre le triplet <Environnement, Plateforme, Utilisateur>.

Notre méta-modèle de contexte résulte de l'analyse et du raffinement de différents méta-modèles de profil et de contexte déjà proposés tels que celui de [15]. La figure 6 montre que notre proposition permet la description de situations contextuelles, composées de trois éléments de contexte.

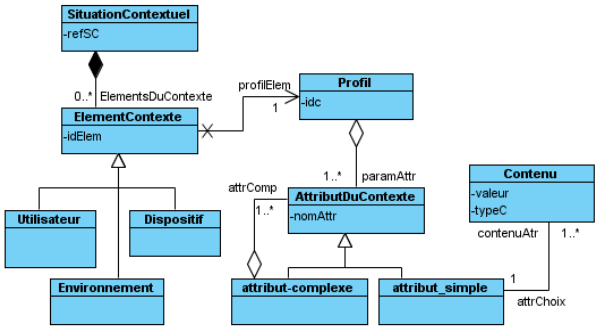


Figure 6 : Méta-modèle de contexte générique

Le concept de contexte est modélisé par la classe *SituationContextuelle*, une classe composite d'éléments représentés par la classe *Element-Contexte*. On notera que chaque élément est lié à un profil. Un pr*ofil* est une composition de descriptions de paramètres ou attributs du contexte. La classe A*ttributDuContexte* décrit des informations caractérisant le profil (par exemple, les compétences et les préférences pour un profil utilisateur). Les attributs du contexte sont soit des *attributSimple* (élémentaires et auxquels des contenus sont associés) (par exemple, l'attribut « Langue ») ou des *attributComplexe* (composés d'autres attributs) (par exemple, l'attribut «Preferences »). L'association entre *Contenu* et *attributSimple* signifie qu'un attribut simple peut être associé à une ou plusieurs valeurs.

Notons la présence de l'attribut AS*Actif* relatif à la classe *attributSimple*. Cet attribut traduit si l'*attribut-Simple* en question doit être pris en compte dans l'adaptation en cours ou non. De même l'attribut *CActif* relatif à la classe *Contenu* mentionne si les variantes qui lui sont associées font partie de la réduction en cours. Par exemple pour la situation contextuelle 0 (cf. figure 4) le contenu Anglais est actif, la tâche spécifier en anglais qui lui est associé fait donc partie de la réduction (cf. Figure 5).

L'objectif de cette modélisation est de pouvoir instancier différents profils (ayant différents paramètres ou attributs, spécifiques à divers contextes), à partir d'un même méta-modèle générique de contexte. Cette structure ne s'applique donc pas à un cadre prédéfini. C'est à chaque application d'instancier cette structure, tout en définissant les éléments et profils correspondants composés d'attributs des contextes. Les attributs de contextes et leurs valeurs possibles (i.e. Contenu) seront utiles par la suite dans la spécification de l'enchainement des tâches spécifiques à chaque interaction.

Méta-Modèle de Tâches Adaptable au Contexte

Les modèles de tâches que nous proposons regroupent des tâches communes à différentes situations contextuelles, et d'autres variables selon le contexte d'utilisation. Cette variabilité est due à la différence de contenus des attributs du contexte, changeant d'une situation contextuelle à une autre. Les tâches communes ne sont pas à adapter, contrairement aux tâches variables. Nous introduisons donc le concept de variabilité dans le modèle de tâches.

Modélisation de la Variabilité dans les Modèles de Tâches. Nous proposons pour l'intégration de la variabilté dans les méta-modèles de tâches, d'étendre le méta-modèle de tâches sans variantes proposé par [13]. Comme pour ce méta-modèle, les tâches peuvent être décorées par un opérateur unaire tel que l'itération (*OpUnaire)* et décomposées en sous-tâches (*OpDecompose)*. Le diagramme de la figure 7 présente le nouveau méta-modèle proposé.

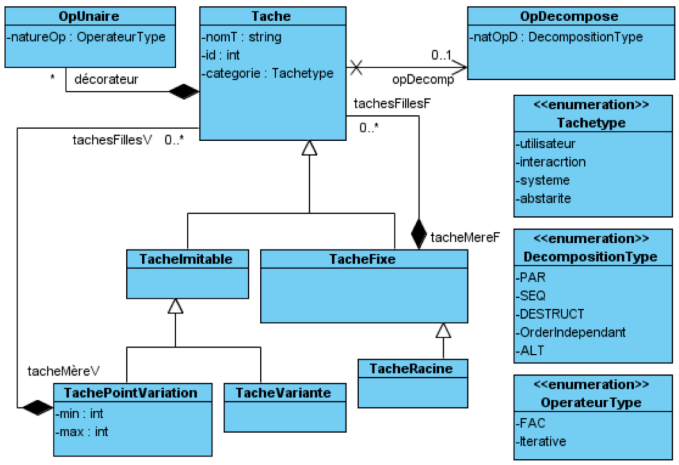


Figure 7 : Méta-modèle de tâches à variantes

Nous distinguons des *TacheFixes* (i.e. toujours faites de la même manière) et des *TacheImitables* (i.e. sensibles au contexte). La classe *TacheImitable* est une abstraction de deux classes : *TachePointVariation* (qui représente une variante selon le contexte) et *TacheVariante* (qui représente une exécution spécifique d'un point de variation). La classe *TacheRacine* représente la tâche racine du modèle de tâches.

La classe *TachePointVariation* est dotée des attributs min et max qui expriment le type de variabilité [1] [17]. Nous utilisons ici les multiplicités introduites dans [9] pour exprimer les 4 types de variabilité :

0..1/1 → Un *point de variation optionnel* (i.e. de type « Option ») représente une tâche pouvant être sélectionnée ou non selon le contexte. Par exemple, la tâche « migration de carte » est un point de variation optionnel.

1..1/n → Un *point de variation alternatif* permet le choix d'une seule variante composante fixe ou imitable parmi n choix possibles. Si le paramètre de contexte duquel dépend la tâche point de variation est actif, une des variantes est sélectionnée, sinon c'est la tâche fixe (tâche par défaut) qui sera sélectionnée. Par exemple, la tâche « spécifier demande » est une tâche *point de variation* alternatif dépendante de l'attribut de contexte « Langue ».

0..1/n → Un *point de variation alternatif optionnel* représente une tâche décomposable en plusieurs tâches imitables. Il s'agit d'une tâche optionnelle, qui peut ne pas être sélectionnée si son contexte d'utilisation n'est pas actif. Cependant, une fois prise en compte, elle admet différentes alternatives.

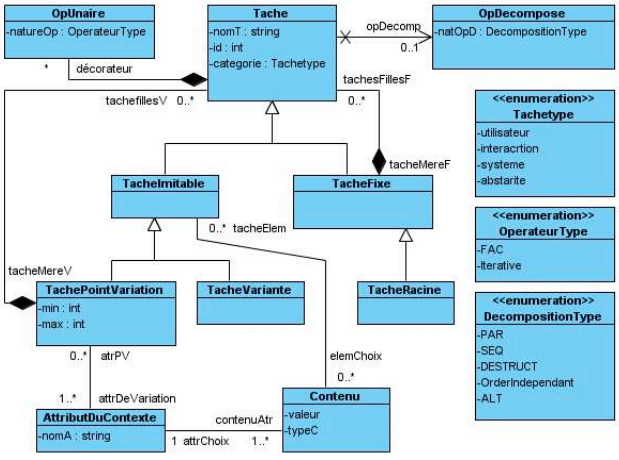
1..n/n → Un *point de variation ensemble d'alternatives* modélise une tâche décomposable en plusieurs variantes, dont plusieurs peuvent coexister.

Les classes *TachePointVariation* et *TacheFixe* représentent des tâches composites de sous-tâches, liées par un opérateur *OpDecompose*. Cet opérateur est toujours de type Alternatif (*ALT*), pour les instances de la classe *TâchePointVariation*. Il peut prendre cependant n'importe quelle valeur (voir *DecompositionType*) pour la classe *TâcheFixe*.

Ce méta-modèle nous permet de distinguer les différents types de variations, pour permettre une adaptation facile et indépendante de chaque type par des règles de transformation génériques. Reste maintenant à assurer le lien entre les tâches imitables et les paramètres du contexte pour diriger leur adaptation.

Augmentation du Méta-Modèle de Tâches à Variantes Par le Contexte. Comme nous l'avons déjà signalé, un système sensible au contexte nécessite de représenter les caractéristiques des contextes qui lui sont associés. Pour ce faire, le méta-modèle de tâches à variantes doit être relié au méta-modèle de contexte.

La figure 8 illustre notre proposition de rapprochement, soit le méta-modèle de tâches adaptable au contexte. La classe *AttributDuContexte* précise un attribut du contexte clé qui fait de la tâche un *point de variation*. La classe *Contenu* liée aux tâches imitables permet de faire le choix entre les différentes alternatives (sous-tâches imitables) d'une tâche *point*



de variation.

Figure 8 : Méta-modèle de tâches adaptable au contexte

Le méta-modèle de tâches adaptable au contexte est donc un ensemble de tâches et d'attributs de contextes non prédéfinis, liés aux tâches point de variation. Une tâche *point de variation* est liée au moins à un attribut du contexte. Ces derniers (instances de *AttributDuContexte*) peuvent être

l'objet de plusieurs tâches *point de variation*, ce qui explique leurs contenus multiples et possibles, spécifiques chacun à un *point de variation*. Le choix d'alternatives d'un point de variation est facilité par ces contenus, objets des tests identifiant les situations contextuelles en cours. Les modèles de tâches instances de ce méta-modèle de tâches adaptable au contexte, seront ainsi faciles à adapter car on peut exprimer naturellement les points de variation caractérisés par leurs contextes.

Après avoir représenté la variabilité et intégré la notion de contexte sur les modèles de tâches, ceux-ci deviennent adaptables pour n'importe quelle application, et ses divers contextes d'utilisation. Leur adaptation nécessite cependant une activité de réduction, où il s'agit de choisir les variantes qui coïncident avec le profil de la situation contextuelle en cours, pour finir avec le modèle cible. Ce processus de réduction est mis en œuvre à travers des transformations de modèles (conformément à l'approche IDM) que nous explicitons ci-dessous.

Règles de Transformation Génériques

Maintenant que nous disposons d'un moyen permettant la spécification d'un modèle de tâches à variantes adaptable au contexte, nous nous intéressons à la transformation d'un tel modèle vers des modèles liés à des situations particulières. Notre objectif est de définir un module d'adaptation pouvant s'appliquer à toute transformation d'un modèle de tâches adaptable au contexte vers un modèle sans variantes. Pour ce faire, nous avons développé des règles de transformation spécifiques à chaque type de point de variation.

La première règle démarre la transformation par la réduction de la tâche racine fixe, et de ses tâches filles. Pour les tâches point de variation, l'identification des variantes à conserver est possible grâce à la mise en correspondance explicite entre les paramètres et contenus liés aux tâches imitables, d'une part, et les caractéristiques des éléments du contexte de la situation en cours, d'autre part.

Les règles de transformation ont été implémentées en ATL [16]. La figure 9 montre un test type à réaliser lors de la réduction de la variabilité. Ce filtre permet de savoir si les attributs de contexte auxquels la tâche est liée (*atrPV*) sont actifs ou non, dans la situation en cours (*atrC*). Il s'applique sur une tâche source *TachePointVariation* (cf. Figure 10 en bas).

```
helper context MMTV!TachePointVariation def : ContexteActif( ) : Boolean =
  let atrC: Sequence (MMCtxt!attributSimple) = MMCtxt!attributSimple.allInstances()
  in let atrPv : Sequence (MMTV!AttributContexte)= self.attrDeVariation
    in atrPv -> forAll (atPv | atrC -> exists (atC | atC.nomAttr = atPv.nomAttr)) ;
```

Figure 9 : Exemple de filtre sur les tâches sensibles au contexte

Selon le type de la tâche point de variation et la valeur de retour du filtre, un traitement correspondant est exécuté. Par exemple, pour la tâche point de variation optionnelle « migration de carte », si le contexte qui lui est associé n'est pas actif (i.e. pas de contrainte sur la taille du dispositif) cette tâche ne fait pas partie de la réduction, sinon on la transforme vers le modèle cible.

```
--cree la tache racine de l'arbre
rule regleRacine {
 from TF: MMTV!TacheFixe  (TF.Racine())
 to
      T: MMT!Tache {
         nom <-  TF.nomT,
         nature <-  TF.categorie,
         operateurDecoration <-  Decos),

      Decos: distinct MMT!Decoration foreach (e in  TF.decorateur) {
              opUn <-  e.natureOp)

    do{
        if (not TF.Feuille())
        T.operateurDecomposition <-  thisModule.CROpDecompTFNPPV(TF);
        else
        OclUndefined;
    }
}

--cree operateur decomposition de tache racine fixe
lazy rule CROpDecompTFNPPV{
    from TF : MMTV!TacheFixe
    to OpN: MMT!OperateurNaire{
     natureOp <-   TF.opDecomp.natOpD,
     tacheMere <-  TF,
     tachesFilles <- Sequence()-> append (TF.tachesFillesF ->collect(t |
        if t.isAlt()
        then
            if   not t.ContexteActif()
            then thisModule.regleFixe(t.filleFixe()->first())
            else if t.AlternativeVariante()
            then thisModule.regleVariante(t.filleVariante()->first())
            else
            endif
```

Figure 10 : Exemple de règle de transformation en ATL

La partie en bas de la figure 10 montre que la transformation est bien définie au niveau méta-modèle. Cette règle permet la transformation d'une tâche fixe du modèle source en une tâche du modèle cible. La partie en haut de la figure fait partie du code exécuté pour générer les tâches filles d'une *TâcheFixe* sélectionnée. Par exemple, pour transformer la tâche Alternative « spécifier demande » si le contexte n'est pas actif (i.e. pas de contraintes sur la langue) la tâche par défaut (langue : Français) remplace cette dernière dans le modèle cible. Sinon la tâche qui correspond au contenu de la situation contextuelle est celle qui remplace la tâche point de variation.

Ainsi, les règles de transformation sont génériques. Le module de transformation s'applique donc sur tous les modèles de tâches de la même manière car il est exprimé au niveau du méta-modèle de tâches. Face à une nouvelle application ou à de nouveaux besoins d'adaptation, une définition d'un modèle de tâches avec variantes suffira.

CONCLUSION ET PERSPECTIVES

La contribution principale de cet article est la proposition d'une nouvelle approche générique basée sur l'IDM et la variabilité pour l'adaptation des modèles de tâches au contexte d'utilisation. Elle a permis de définir des règles de transformations génériques qui permettent aux concepteurs d'interfaces adaptables de définir seulement un modèle de tâches adaptables et le contexte d'usage.

Nous avons réalisé une expérience utilisateur afin d'évaluer la facilité de la mise en œuvre des modèles de tâches avec variabilité. Les concepteurs ont alors apprécié la bonne structuration du modèle face au contexte et l'expression des contraintes de choix à travers les multiplicités. Néanmoins ils ont aussi noté le manque de contraintes entre les différentes alternatives du modèle et la complexité du modèle face à un contexte important. Ce sont ces deux points qui nous devons approfondir en premier lieu.

Il nous semble aussi nécessaire de mettre en œuvre une démarche qui permettra d'aller plus loin pour faire la transformation du modèle résultant vers du code. Cela nécessitera un développement d'un deuxième module de transformation tel que celui de [13]. Il sera ainsi possible d'adapter l'interface lors de l'exécution du système rendant ainsi le modèle de tâches avec variantes vivant à l'exécution. Nous espérons ainsi aboutir à des systèmes interactifs adaptables qui conservent au mieux leur utilisabilité.

REMERCIEMENTS

Ce travail a été partiellement financé par le projet ITEA UsiXML.

BIBLIOGRAPHIE

1. Bachmann F., Bass L., Managing variability in software architecture, ACM SIGSOFT Software Engineering Notes, Vol. 26, n°3, 2001, pp. 126-132.
2. Balme L., Interfaces homme-machine plastiques: Une approche par composants dynamiques, Thèse de Doctorat, Université Joseph Fourier, 2008.
3. Calvary G. Coutaz J., Métamorphose des IHM et Plasticité, In *Revue d'Interaction Homme-Machine (RIHM)*. David, B., Kolski, C. (Eds), Europia (Publ.), Volume 8, Numéro 1, ISSN 1289-2963, 2007, pp. 35-60.
4. Chaari T., Laforest F., Flory A., Adaptation des applications au contexte en utilisant les services Web, Coutaz, J., Lecomte, S.(Eds.), In *UbiMob'05*, Grenoble, France, 2005, pp. 111-118.
5. Czarnecki K., Eisenecker U.W., Generative Programming – Methods, Tools and Applications, 2000.
6. Favre J.-M., Towards a basic theory to model-driven engineering, In *3rd Workshop in Software Model Engineering 2004*, joint event with UML2004, Lisboa, Portugal, 2004, pp. 9-17.
7. Ganneau V., Calvary G., Rachel D., Métamodèle de Règles d'Adaptation pour la Plasticité des Interfaces Homme-Machine, In *IHM'2007*. Paris, France, 2007, pp 91-98.
8. Van Gurp Jf., Bosch J., Svahnberg M., Managing Variability in Software Product Lines, Landelijk Architectuur Congres, Amsterdam, 2000.
9. Matthias R., Detlef S., Ilian P.; Modeling Variability for Object-Oriented Product Lines, In A. Buchmann, F. Buschmann (Eds.): *ECOOP 2003 Workshop Reader*. Springer, LNCS, 2003.
10. Paternò, F., Santoro, C., One Model, Many Interfaces, In *Proc. of CADUI'2002*, Kluwer Academics Pub., Valenciennes, France, May 2002.
11. Pribeanu, C., Vanderdonckt, J., Limbourg, Q., Task Modelling for Context Sensitive User Interfaces, In *Proc. of DSV-IS'2001*, LNCS 2220, Springer-Verlag, Berlin, 2001, pp. 49-68.
12. Samaan K, Tarpin-Bernard F., Task models and interaction models in a multiple user interfaces generation process. *In Proc. of TAMODIA '04*, République Tchèque, 2004, pp. 137-144.
13. Sottet JS., Mega-IHM, Malléabilité des interfaces homme-machine dirigée par les modèles, Thèse de Doctorat, Université Joseph Fourier, 2008.
14. Souchon, N., Limbourg, Q., Vanderdonckt, J., Task Modelling in Multiple Contexts of Use, In *Proc. of DSV-IS'2002*, LNCS. 2545, Springer-Verlag, Berlin, 2002, pp. 59-73.
15. Tchienehom P., Modèle générique de profils pour la personnalisation de l'accès à l'information, In *INFORSID'2005*, 2005, pp 269-284.
16. The Eclipse Foundation, Wiki ATL, http://wiki.eclipse.org/ATL/User_Guide. Dernière visite : Mai 2009
17. VanderMaβen T., Lichter H., Modeling variability by UML use case diagrams, In *International Workshop on Requirements Engineering for Product Lines (REPL'02)*, 2002, pp. 19-25.
18. Van Gurp J., Bosch J., Svahnberg M., On the notion of variability in software Product Lines, In *Proc. of the Working IEEE/IFIP Conference on Software Architecture*, 2001, pp. 45-54.